Student Name: **Stud id:** **sect #:** **serial#:**

University of Bahrain College of Information Technology
Department of Computer Science

ITCS332: Organization of Programming Languages **Quiz #6: Assignment + Subprograms**

****

1) The parameters written in the subprogram header are called *formal*, and the parameters written in the subprogram call are *actual*.

2) Name two thing placed on the activation record by the caller subprogram: *return address* and *parameters*.

3) In expressions, the side effects may occur when a function changes *non-local vaiables* or *2-way parameters.*

4) The format and the structure of the activation record are decided by *Language Implementer*. The nesting of subprogram definitions is decided by *Language designer.*

5) A Procedure abstracts a program unit into *a standalone statement*; while a function abstracts a program unit into *an expression (part of a statement).*

6) In programming languages, the actual parameters are bound to formal parameters according to their *position* or *names*.

Carefully study the following C-like code and answer the next 2 questions

```
int fun (int *k)
{
  *k += 5 ;
  return 2 * (*k) - 5 ;
}
void main()
{
    int i = 6, j = 4, res1, res2 ;
    res1 = ( i / 2 ) + fun ( &i);
    res2 = fun (&j ) + (j / 2);
}
```

7) What are the values of `res1`and `res2` if the operands in the expressions are evaluated **left to right**?

res1 = *i/2 + fun(&i)*
= *6/2 + fun(6)*
= *3 + 2*11 – 5*
= *3 + 17*
= *20*

res2 = *fun(&j) + j/2*
= *fun(4) + j/2*
= *2*9 – 5 + 9/2*
= *13 + 4*
= *17*

Student Name: **Stud id:** **sect #:** **serial#:**

University of Bahrain College of Information Technology
Department of Computer Science

ITCS332: Organization of Programming Languages **Quiz #6: Assignment + Subprograms**

****

8) In PHP, the expression "6" == 6 produces *true*, while the expression "6" === 6 6produces *false*

9) In programming languages, the actual parameters are bound to formal parameters according to their *position* or *names*.

10) The evaluation of any expression consists of evaluating *Operators* and evaluating *Operands.*

11) The two important considerations in choosing parameter-passing methods are: *efficiency* and *whether One-way or two-way* .

12) The nesting of subprogram definitions is decided by *Language designer.* The format and the structure of the activation record are specified by *Language Implementer*.

13) In passing a matrix as a parameter, C++ requires specifying *the number of columns* needed in generating the access function. The parameter passing methods are implemented by using *the run-time stack*.

Carefully study the following C-like code and answer the next question.

```
int fun (int *k)
{
  *k += 8 ;
  return 2 * (*k) - 9 ;
}
void main()
{
    int i = 6, j = 4, tot1, tot2 ;
    tot1= ( i / 2 ) + fun ( &i);
    tot2 = fun (&j ) + (j / 2);
}
```

14) What are the values of `tot1`and `tot2` if the operands in the expressions are evaluated **left to right**?

tot1 = i/2 + fun(&i)
= 6/2 + fun(6)
*= 3 + 2*14-9*
= 3 + 19
= 22

tot2 = fun(&j) + j/2
= fun(4) + j/2
*= 2*12-9 + 12/2*
= 15 + 6
= 21